

# 242ine

Issue #5

August 2018



Published by:

Timothy Swenson  
swenson\_t@sbcglobal.net  
swensont@lanset.com

ZXzine is published as a service to the Sinclair ZX81 community. Writers are invited to submit articles for publication. Readers are invited to submit article ideas.

Created using Open Source Tools:

- OpenOffice
- Scribus
- Gimp
- SZ81
- EightyOne

Copyright 2018  
Timothy Swenson

Creative Commons License

- Attribution
- Non-Commercial
- Share-Alike

You are free:

- To copy, distribute, display, and perform the work.
- To make derivative works.
- To redistribute the work.

**Editorial** ..... 1

**Adding Assembly to Basic** ..... 1

**Loading a File into Memory** ..... 2

**SoundPlayers** ..... 3

**ZX81 Resources at the Internet Archive** ..... 5

**JtyOne** ..... 6

**Time Designs Magazine** ..... 7

**Z80asm** ..... 8

**Toby Radloff** ..... 9

## Editorial

I guess it does take a year to create enough material for another issue of ZXzine. Well, I knew that I was not making any promises as to release dates when I started this e-zine.

There was one project that I was working on that was too long to be published in ZXzine, so I posted it to my Google Sites page. I was exploring the ZonX-81 sound system as emulated on the sz81 emulator. If you look in the Google Sites page where this e-zine is hosted, look for ZonX.pdf. I collected a number of older articles on the ZonX and condensed it down into a single source. There are examples of programming the ZonX with BASIC and with C (using Z88dk). There is a lot more to explore with the ZonX, but this document is a good start.

I have already started work on articles for the next ZXzine, so I hope it won't be another 12 months before I finish them.

## Adding Assembly to Basic

In my programming so far, I've done just pure BASIC using zxtxt2p, or just assembly using Pasm0. The problem I've faced is doing the combination of assembly in a REM statement with BASIC. In the past the process was to create a REM statement and type in a hex loader program. Use the hex loader to load the assembly (converted to hex) into the REM statement. Then, the hex loader had to be deleted one line at a time, and the actual program could be typed in.

Zxtxt2p does not allow for inline assembly, and Pasm0 does not allow for inline BASIC. There had to be a way to mix the two. After some thinking, I knew that I needed to find a way to take some assembly in binary format and inject it into the .P file created by zxtxt2p, within the normal REM statement.

My main reason for doing this that sz81 emulates the Zon X-81 sound device. To use this device, a small bit of assembly code is needed.

I thought about writing a C program that would do something like this, but after chatting with Graeme Gregory, he suggested using the Unix/Linux tool dd. dd is utility that reads data from one source (file or device) and outputs it to another source (again, file or device). It can do conversion on the data. It can read or write with a certain block size, etc. Looking in the dd man page I found two features that are what I need, conv=notruc and seek.

The option conv=notruc allows dd to overwrite existing data in the output file. The seek option tell dd to read a certain amount of blocks into the file before doing the write. By using a block size of 1 byte, the seek command can read X bytes into a file.

Here is the process to merge assembly and BASIC.

1. Write the assembly code. Make sure to leave out any include statements that normally create a .P file. Here is the code in my example:

```
;; zon1.asm

LD      A,0
OUT     (223),A
LD      A,0
OUT     (31),A
RET
```

I compiled this with Pasm0 with the following command line:

```
% pasmo zon1.asm zon1.bin
```

2. Create the BASIC program. Make sure to create a REM statement with enough characters for the assembly. To determine the size of the assembly program, run the 'ls -l' on the file and it will show the number of bytes:

```
-rw-rw-r-- 1 swensont swensont 9 Jul 19 20:01
zon1.bin
```

Here is the BASIC that I created using the example from the Zon X-81 manual.

```
1 rem XXXXXXXXXXXX
2 goto 7
3 poke a,d
4 poke b,c
```



```

5 let x = usr e
6 return
7 let a = 16515
8 let b = 16519
9 let e = 16514
10 rem random input
20 let c = int ( rnd * 255 )
30 let d = int ( rnd * 14 )
40 gosub 3
50 for t = 1 to 30
60 next t
70 goto 20

```

I then ran test1.bas through zxtxt2p to create a .P file:

```
% zxtxt2p -o test1.P test1.bas
```

3. Now to use dd to add the zon1.bin file into the test1.P file:

```
% dd conv=notrunc seek=121 bs=1
if=zon1.bin of=test1.P
```

I looked at the .P file using 'od -h' and found where the REM statement was. I found how far into the file it was and used that for the value of the seek option. My count was off because the first time I did this I had some extra X's in the REM statement before the code. I adjusted the seek number and found the right number. If using zxtxt2p, this above line should work for you. If some other tool, you might need to experiment a bit. To make sure that all was working, I made sure that test1.P worked.

## Loading a File into Memory

The sz81 emulator, from at least version 2.3.6, includes a new version of the LOAD command. The version loads a file into memory at a specified location. This is similar to how to the Spectrum or T/S 2068 version of LOAD. An example is:

```
LOAD "file.txt; 32768"
```

The command will read in the file called "file.txt" and load it into memory at location 32768. From there the data is accessible by any program that you

write. The file is loaded directly, so the file can be a machine code program, a chunk of data, or a text file. By using PEEK, it is possible to read the data in memory, one byte at a time.

I was interested in using the feature to load in a text file and then use the ZX81 to process the data. It is simple to create a small database in a text file using the colon (:) as a field delimiter, the "new line" character (ASCII 10) as an end of record marker, and the AT sign (@) as an end of file marker. An example is this:

```

ABC:123
DEF:456
@

```

Before going too far, I had to know exactly how the characters from the text file were stored in memory. Was there a conversion from ASCII to the ZX81 character set? I created a simple test program that would load a text file and then output the decimal value of each character loaded.

```

load "load.txt; 32768"
print "loaded file"
print

let mem = 32767

for x = 1 to 17
  let a = peek(mem + x)
  print a;" ";chr$(a)
next x

```

I ran this program on the above example text file. The result is that the data is stored exactly as it is seen in the text file, with no conversion to the ZX81 character set (which is what I thought). So, I would have to convert the characters to be useful for the ZX81.

To go beyond the test program, I wrote some code that will read in a field and return it to the calling program. The calling program must know if the field is character or number based. The number can then be converted from text to a value.

Here is the example "database", which contains a record with a field for an expense type and a field



for the value of the expense. The example program will read in a number of records until it reaches the "end of file" marker. A final total for the expenses will be printed out. The routine returns a value if the "end of record" or "end of file" marker is reached. In my sample code, I don't need the "end of record", but it is there so it is possible to read in fields from a database with an unknown set of fields.

```
# load.bas

# load the record file
  load "load.txt; 32768"
  print "loaded file"
  print
  let mem = 32768
  let aa = 0
  let total = 0

# get the first field (should be
text)
@loop:
  gosub @loadnext
  if aa = 1 then goto @done
  print "      ";a$;"      ";
# get the second field (should be
number)
  gosub @loadnext
  let t = val(a$)
  let total = total + t
  print t
  goto @loop

@done:
  print "Total : ";total
  stop

# subroutine loadnext
# Will keep reading memory until
and end of record
@loadnext:
  let a$ = ""
@loadloop:
  let a = peek(mem)
  let mem = mem + 1
  if a = 64 then goto @enddone
  if a = 58 then goto @endrecord
  if a = 10 then goto @endfield
# found a letter
```

```
  if a >= 65 and a <= 90 then let
a$ = a$ + chr$(a-27)
# found a number
  if a <= 57 and a >= 48 then let
a$ = a$ + chr$(a-20)
  goto @loadloop
@enddone:
# aa has value of 1 if end of file

  let aa = 1
  return
@endrecord:
# aa has value of 2 if end of
record
  let aa = 2
  return
@endfield:
  return
```

The text database:

```
GAS:10
UTIL:13
FOOD:20
@
```

## Sound Players

Since sz81 supports the emulation of the Zon-X81 sound system, I have been looking to see what programs use the Zon-X81 features. Most discussion of these programs were found on the Sinclair ZX80 / ZX81 Forums at [sinclairzxworld.com](http://sinclairzxworld.com).

There are two formats for the sound players to use, STC and PT3. STC is the format used by "Sound Tracker" a program for the Spectrum. After some research, I could not find the origin of the PT3 file format.

### Juk.p

This player, written by Andy Rea and plays STC files. It finds all of the STC files in a local directory and starts playing them, one at a time. The only controls are the N key for Next song and the P key for Previous song. When using the

```
SONG 1,NEXT IN 7 TICKS
```

player, I had to hold either key down for a few seconds for the player to sense the key. While playing the song, the text "song X, next in X ticks" if flashed at random locations on the screen. That can get a little annoying after a while.

<https://sinclairzxworld.com/download/file.php?id=3272>

### stcplayer.p

This player of STC files was also written by Andy Rea, based on code from Sergy Bulba. When the

```
STC PLAYER FOR ZX81
ORIGINAL CODE SERGY BULBA
PORTED BY ANDY REA
FILENAME ?
LOADED 02.STC;32768
PRESS NEWLINE TO START OVER
```

```
"█"
```

program starts, it prompts for a file name. If the file "FOO.STC" exists in the local directory, then you will only need to type "FOO". Since the ZX81 only does upper case letters, all files will need to be in uppercase, including the file extension. Once the player starts, you can hit enter to stop it playing. The program will then prompt for another file name.

<https://www.sinclairzxworld.com/download/file.php?id=1011>

### ui-player.p

This player, written by Siggi from the ZX81/ZX80 Forums, plays PT3 files. Once it loads, type RUN to start the program. Use the L key to load a file.

```
START PLAYER LOAD SONG
```

The program expects the full file name, ie. FOO.PT3. Once the file is loaded, then press S to start playing the song. While the song is playing,

```
STOP PLAYER PAUSE
PLAYING POPCORN.PT3
```

The S key will stop the the song from being played, or the P key will pause the song.

<https://www.sinclairzxworld.com/download/file.php?id=474>

## bigstc

This is a large collection of STC files (over 1000) with the Juk.p player to play them. It was collected by a user on the Sinclair ZX81 forums called Moggy.

<https://sinclairzxworld.com/download/file.php?id=4338>

## ZX81 Resources at the Internet Archive

Founded in 1996, the Internet Archive has been a place to store electronic material for long term archival. It has allowed the average person to upload material to the archive. While doing some research,

Sync - This was the first magazine for the Sinclair computers. It started as a magazine for the ZX80, then added the ZX81 and then the T/S 2068.



Timex Sinclair User - This was the second magazine for Sinclair computers. It came out after Timex bought the marketing rights to Sinclair computers in the US, hence the Timex Sinclair in the name. It covered the T/S 1000, T/S 1500, and the T/S 2068.

Ramblings - The Official Timex Computer Club

Newsletter, published by Timex Computer Corporation.

Syntax - ZX80 - This is the oldest Sinclair newsletter in the US. It is a publication of The Harvard Group of Boston, MA. It started publishing in 1980.

Synchro-Sette - Paper and cassette based newsletter for the ZX81, started in April 1982.

Update - This newsletter started as "T/S 2068 Update" out of Florida in 1988. Frank Davis of Indiana took it over and changed it from quarterly to bi-monthly. He expanded the coverage to all Sinclair computers.

Time Designs - Tim Woods started this newsletter in 1985 and it covered all Sinclair computers.

T.S. Horizons - Sinclair newsletter started in November 1983 by Rick Duncan.

SincLink - Newsletter of the Toronto Timex/Sinclair User Group.

Sincus News - The newsletter of Sinclair Computer Users Society. Started publishing in 1982.

SincWare News - The Journal for electronic and other technical applications for T/S Computers. Published by Tom B. Woods.

Nite Times News - Newsletter of the Chicago Timex Sinclair User Group.

QuarTerS - Timex Sinclair newsletter published by WMJ Data Systems, a T/S vendor.

Zir Qlive Alive - Newsletter for the North American Timex Sinclair User Group. When local user groups started folding, a number of user groups joined together to form a single group for the US and Canada. It started in the spring of 1991.

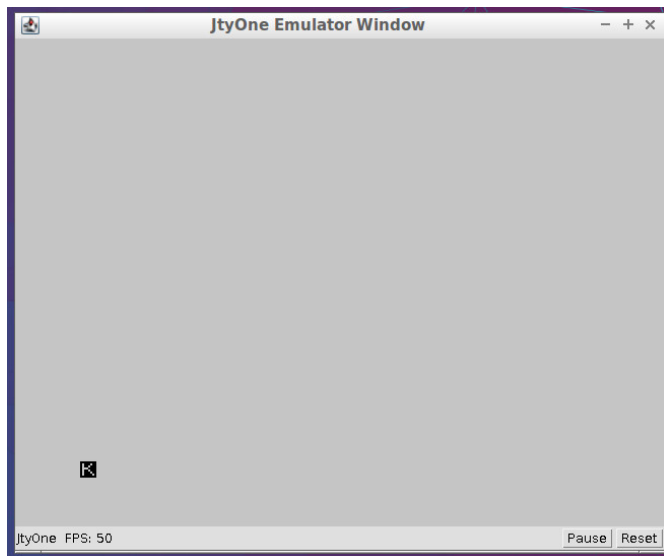
The Plotter - Newsletter of the Clackamas Computer Applied Training Society, a user group formed to support Timex/Sinclair users in Oregon.



The Best of SUM - Compilation of the best articles from the S.U.M. newsletter, from the Gainesville, Florida Timex Sinclair User Group.

## JtyOne

Most ZX81 emulators are designed for either Windows or Linux. If you are moving across platforms, your emulator of choice may not move with you. One emulator, JtyOne, was written in Java, so that it is portable across any computer that will run Java. Simon Holdsworth wrote JtyOne



version 1.0 back in 2006. This version of JtyOne can be run as a stand-alone application. There is an updated version, 1.1, but it will not run as a standalone application.

The emulator can be downloaded from this link:  
<http://www.zx81stuff.org.uk/zx81/jtyone.jar>

Once downloaded, the whole emulator is in a single .jar file, which is 110K in size. There is no installation or configuration. To start the emulator, from Windows, double click on the .jar file. In Linux, you'll need to run it from the command line;

```
% java -jar jtyone.jar
```

This will fire up a Window that has two buttons, one to pause the emulator and one to reset the emulator. There are no

other menu options.

To load the emulator with a program, just add the name of the program file at the end of the command line:

```
% java -jar jtyone.jar prog.tzx
```

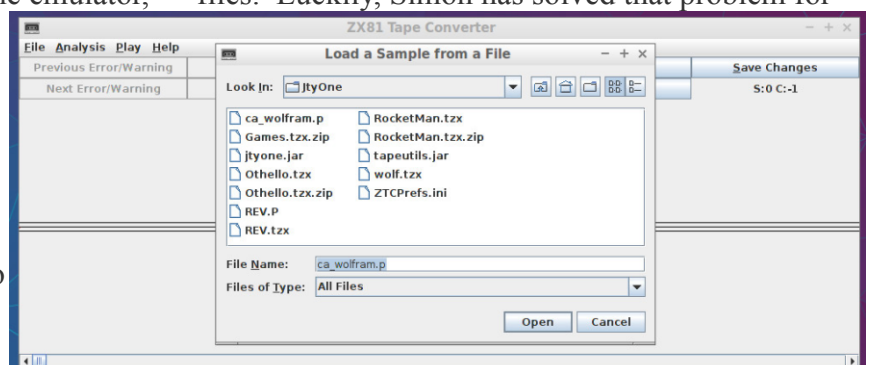
The emulator window will open, wait a few seconds and LOAD "" as if typed by magic, and the program will load. There is no graphic of the ZX81 keyboard to pop up in the emulator, so you'll have to know your way around the ZX81 keyboard.

One issue with JtyOne is that it will only load .tzx files. It will not load .P or .81 files. Simon also has on his website an archive of ZX81 programs that are stored in .tzx format and can be downloaded. Just go to [www.zx81stuff.org.uk](http://www.zx81stuff.org.uk) to access the archive. There are enough programs there to keep you entertained for months.

Using the standard LOAD and SAVE commands seem to call the same routine used for saving to tape, so you can't load or save using just the command line. With no menu options, there does not appear to be a way to save files from the emulator.

I have tested JtyOne with a number of programs including those created with the Pasm0 assembler, with Z88dk and zxtexp2. All of them run fine on JtyOne. I tested with a Pseduo Hi-Res program and that worked, but a true Hi-Res program did not work. The Hi-Res needs a hardware modification on a real ZX81, so JtyOne does not emulate that.

Since most of the tools these days create binary files in .P, only loading .tzx files can be a bit of a problem. The issue is how to convert .P to .tzx files. Luckily, Simon has solved that problem for



us. Also on this website is a tape utility that converts zx81 audio tapes to .P or .tZX. The program, called tapeutils, is also Java based and portable. It can be found on Simon's website:

<http://www.zx81stuff.org.uk/zx81/tapeutils/overview.html>

From the overview page, it is not obvious how to convert between the two formats. Execute the tapeutils.jar file in the normal way. Once the window pops up, use the GUI (File -> Open) to load a .P file. Once the file is loaded then, select File -> Save As TZX. There does not appear to be a way to do this via the command line, which would have helped in automating the conversion.

The tape utility also has the ability to show the listing of the program, which pops up in another window.

JtyOne is a nice, lightweight, simple ZX81 emulator that has a very small footprint on disk and I highly recommend it.

## Time Designs Magazine

When Timex Computer Corporation abandoned the home computer market, most everyone thought that that was the end for Sinclair and Timex users. With the close-out sales for the T/S 1000, T/S 1500, and T/S 2068, the number of users grew and they were looking for sources of information to use the computer.

The slick paper professional magazines did not last much after Timex Computer Corporation closing down in the US. Sync



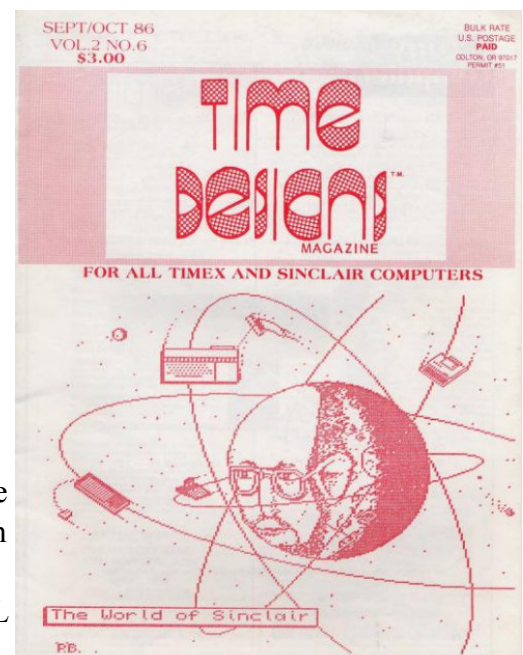
published their last issue with the March/April 1984 issue. Timex Sinclair User magazine only lasted 8 issues ending in December 1983.



A number of newsletters were run by the different user groups. There were a few smaller format publications like T.S Horizons and Update!. Late in 1984 Time Design magazine came out of Colton, Oregon, published by Tim and Stephanie Woods. Time Design has a professional look. It was printed on 11x17 paper and then folded in magazine style. Most issues averaged 42 pages or more. It has some heft to it when reading it.

Most major Sinclair or Timex dealers advertized in the magazine. The Woods would travel to and report on a number of Sinclair shows in the U.S. The magazine covered all Sinclair computers. The very first issue covered the announcement of the Sinclair QL in the US.

It would soon start carrying articles on the QL. When Sinclair announced the sale of his computers to Amstrad, it was detailed in Time Designs and there was an article on the future of the QL in the



US. When the Z88 was announced, Time Designs had a detailed review of it and later reviewed the reviews from the professional computer press. Tim Woods was so impressed with the Z88 that he started publication of the "Club Z88" newsletter.

Time Designs continued published until 1989. The last issue I can find came out in early 1989. I never subscribed because I did not know about the magazine while it was still published, but I have picked up a good collection of them years later. I'm finding them a good source of material for T/S 1000 and T/S 2068 programming and the software sold for these systems.

## z80asm

In looking at other cross assemblers, I realized that z88dk comes with its own cross assembler, z80asm. I thought would try out this cross compiler and see how it differs from Pasm, the other cross compiler that I use for the ZX81. When I first tried out z80am, I could not get a working binary. When I compared the binary with the binary compiled by Pasm, I found that they were the same size, but in one or two different locations, the bytes were different. I asked Graeme Green to take a look. He installed z880asm and did not find the same problem and he was using a newer version. It appeared the older version that I was using had a bug. I updated to the most recent version of Z88dk and now z80am is now working.

In general, the assembly code written for Pasm will work with z80asm, except for a few differences:

1. Most of the mnemonics work between Pasm and early z80asm, but there is one difference:

Pasm	z80asm
DEFB	DEFB
DEFW	DEFW
DEFS	DEFS
ORG	ORG
EQU	DEFC <-- Note the change

3. \$ as an operator

With Pasm, the \$ is also an operator that gives the value of the position counter at the begin of the current sentence. Here is an example:

```
line0:
    DEFB $00,$01      ; line
number
    DEFW line1-$-2     ; line
length
```

With z80asm there is no \$ operator, so the code has to be changes as follows:

```
line0:
    DEFB $00,$01      ; line
number
line01:
    DEFW line1-line01-2 ; line
length
```

4. z80asm is case insensitive while Pasm is case-sensitive

VARs and vars are different with Pasm.

```
VARs:    DEFW vars
DEST:    DEFW 0
E_LINE:  DEFW vars+1
```

With z80asm VARs and vars are the same, so I had to rename one of them.

```
VARs1:   DEFW vars
DEST:    DEFW 0
E_LINE:  DEFW vars+1
```

There is a -nocase option with Pasm to make it case insensitive.

The way to compile with z80am is different than Pasm. The command line is:

```
% z80asm -b file.asm
```

The -b option says to create a binary file. This will end with a .bin. There are three additional files that



are created; .map, .obj, .sym. These can be deleted. To get the program to run in a ZX81 emulator, just do:

```
% mv file.bin file.p
```

to change the extension of the file from .bin to .p.

The z80asm user guide found on the Z88dk site is not very in-depth. When looking for online information on z80asm, remember that there a number of cross compilers also called z80asm, including from SLR Systems from 1984 (which predates this z80asm). The z80asm from Z88dk will say it is copyright from "Interlogic 1997-2003".

I have not seen any difference between z80asm and Pasm0 with the short testing that I have done. If you already have z88dk installed and you want to use a cross-assembler, then there is no need to download and install Pasm0.

## **Toby Radloff**

In the US there were never any famous people that used Timex/Sinclair computers..... except one.

In the late 1980's, MTV was still showing music videos. If you were younger than 30, there as a good chance that you were watching MTV at some point. There was one show, possibly called the "Cutting Edge", that was known for being a little out there. They would show segments that looked like they came from college or a community access cable. These were very low budgets, short segments.

One person that was on one of these segments was Toby Radloff. Toby Radloff worked with Harvey Pekar, a cartoonist who did the "American Splendor" comic, an underground comic. After a while, Harvey would feature Toby in the comic.

The story goes that when MTV came to do a story on Harvey, they met Toby. Toby has a unique voice and speech mannerism that is hard to miss. Somehow the MTV crew took a liking to Toby and used him on a few short features. Eventually, Toby

would have his own part of a local cable access tv show.

Toby sold himself as the "genuine" nerd. He had a very nerdy way of talking and staying focused on a topic. Because of this, he got airtime on MTV, mostly on the "cutting edge" show. Being a Computer Science graduate and being under 30, I was watching MTV and I saw these "genuine nerd" MTV segments. I was not an avid fan, but I know who Toby was and what he looked like.

In 1989, the Capital Area Timex Sincalir (CATS) user group hosted the CATSfest, the annual Sinclair computer show. I had had moved to the Washington D.C. area about a year before the fest and had only recently joined the CATS user group. I attended the CATSfest along with the dinner held the night before the show started.

Toward the end of dinner, there were three people that showed up late. It seemed their drive from Cleveland had taken them longer than they expected. As I look up to see who these gentlemen were, I say to myself, "Holy snikes, that's Toby Radloff"and just stared. As about the only person under 30 at the show (outside of the guys from Cleveland), I was the only one there who know who Toby was. I had a tough time trying to explain to the others who Toby was and why he was moderately famous.

I got a chance to talk with Toby and his companions, Doug Gillespie and a guy named John. It turned out that Toby was a user of the T/S 2068 and moderately active in the Cleveland Timex Sinclair user group.

A few years ago there was a movie on Harvey Pekar, called "American Splendar". Judah Friedlander played the part of Toby and did it rather well. I've found a number of videos on Youtube that Toby has done, but I have yet run across one where he mentions the T/S 2068.